JPA2000-172840

(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **2000172840 A**

(43) Date of publication of application: **23.06.00**

(51) Int. Cl **G06T 5/20**
H04N 5/262

(21) Application number: **11342102**

(22) Date of filing: **05.01.98**

(30) Priority: **02.01.97 US 97 35733**
27.10.97 US 97 957833

(62) Division of application: **10000359**

(71) Applicant: **RUNAWAY TECHNOL INC**

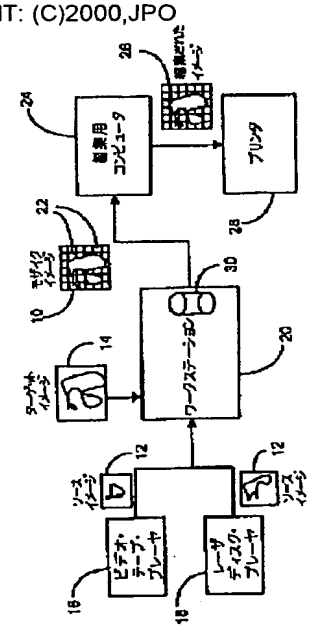
(72) Inventor: **SILVERS ROBERT S**

(54) **DIGITAL COMPOSITION OF MOSAIC IMAGE**

(57) Abstract:

PROBLEM TO BE SOLVED: To generate a mosaic image approximating a target image from a data base for a source image by generating mosaic images having source images which are arranged at the respective tile parts of mosaic images corresponding to respective tile parts and best match them.

SOLUTION: A computer workstation 20 acquires a source image 12 from a video tape player 16 and a laser disk player 18. Further, the computer workstation 20 inputs a target image 14 and generates a mosaic image 10 from the target image 14 and the source image 12. The mosaic image 10 generated by mosaic software includes an array of tiles 22. Each tile 22 is a source image 12 and the whole outward appearance of the mosaic image 10 approximates the outward appearance of the target image 14.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－172840
（Ｐ２０００－１７２８４０Ａ）
(43)公開日　平成12年６月23日(2000.6.23)

(51)Int.Cl.[7]	識別記号	Ｆ Ｉ		テーマコード（参考）
G06T 5/20		G06F 15/68	400 Z	
H04N 5/262		H04N 5/262		

審査請求　未請求　請求項の数14　ＯＬ　（全15頁）

(21)出願番号　特願平11－342102
(62)分割の表示　特願平10－359の分割
(22)出願日　平成10年１月５日(1998.1.5)

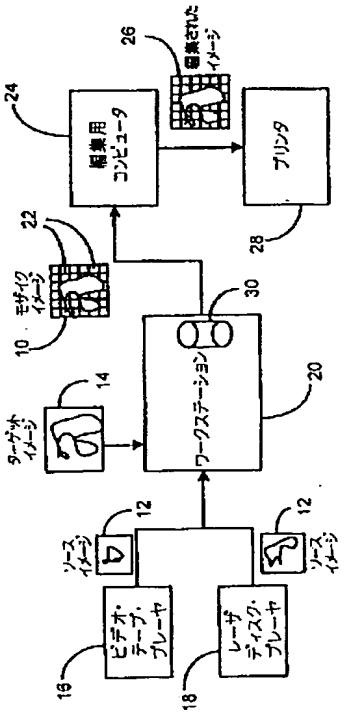
(31)優先権主張番号　６０／０３５７３３
(32)優先日　平成９年１月２日(1997.1.2)
(33)優先権主張国　米国（ＵＳ）
(31)優先権主張番号　０８／９５７８３３
(32)優先日　平成９年10月27日(1997.10.27)
(33)優先権主張国　米国（ＵＳ）

(72)発明者　ロバート・エス・シルバーズ
アメリカ合衆国マサチューセッツ州02139，ケンブリッジ，フランクリン・ストリート　129，ナンバー　105
(74)代理人　100089705
弁理士　社本　一夫　（外５名）

最終頁に続く

(54)【発明の名称】モザイク・イメージのデジタル・コンポジション

(57)【要約】
【課題】　ソース・イメージのデータベースからのモザイク・イメージの形成。
【解決手段】　モザイク・イメージを生成するために、ソース・イメージが分析され、選択され、編成される。ターゲット・イメージをタイル領域に分割し、自乗平均誤差の平方根を計算することにより、そのそれぞれを個々のソース・イメージの部分と比較し、得られる最も整合性の高いソース・イメージを決定する。それぞれの最もの整合性の高いソース・イメージを各タイル領域に置くことにより、モザイク・イメージを形成する。



1

【特許請求の範囲】

【請求項１】　複数のソース・イメージを用いることによりターゲット・イメージに近似する外観を持つモザイク・イメージを生成するコンピュータ・プログラムを記録した記録媒体であって、

ターゲット・イメージをコンピュータへロードするステップと、

前記ターゲット・イメージを、各々が該ターゲット・イメージの個別の位置を表す複数のタイル領域へ分割するステップとを備え、

各タイル領域ごとに、

視覚的類似性の値を生成するためにソース・イメージをタイル領域と比較する比較ステップであって、各ソース・イメージの複数の部分を分析するステップを含む、比較ステップと、

タイル領域を表すために視覚的類似性の最大の値を持つソース・イメージを選択する選択ステップと、

選択された前記ソース・イメージを、前記モザイク・イメージにおいて、前記タイル領域の位置に対応する位置に配置するステップとを備える方法を行うように機能するコンピュータ・プログラムを記録した記録媒体。

【請求項２】　請求項１記載の記録媒体において、前記コンピュータ・プログラムにより行われる前記方法が、前記タイル領域を、各々が前記ソース・イメージの特定の部分に対応する個々のサブ領域に分割するステップと、視覚的類似性の値を生成するために個々のサブ領域を個々のソース・イメージ部分と比較するステップとを更に含む、記録媒体。

【請求項３】　請求項２記載の記録媒体において、前記コンピュータ・プログラムにより行われる前記方法が、各サブ領域当たり１つのピクセルを有するソース・イメージを用いるステップを更に含む、記録媒体。

【請求項４】　請求項１記載の記録媒体において、前記比較ステップは、赤、緑および青のチャネルの自乗平均誤差の平方根の一形態を計算するステップを更に含む、記録媒体。

【請求項５】　請求項１記載の記録媒体において、前記コンピュータ・プログラムにより行われる前記方法が、１つのソース・イメージがモザイク・イメージにおいて１回より多く現れることのないように、前記選択ステップにおいて選択されたソース・イメージを考慮から除くステップを更に含む、記録媒体。

【請求項６】　請求項１記載の方記録媒体において、前記コンピュータ・プログラムにより行われる前記方法が、ソース・イメージを捕捉する捕捉ステップと、捕捉されたソース・イメージをデータベースに記憶するステップとを更に含む、記録媒体。

【請求項７】　請求項６記載の記録媒体において、前記コンピュータ・プログラムにより行われる前記方法が、前記捕捉ステップにおいて捕捉されたソース・イメージを方形に切り取ることにより、変更されたソース・イメージを生成するステップを含む、記録媒体。

2

【請求項８】　請求項７記載の記録媒体において、前記コンピュータ・プログラムにより行われる前記方法が、風景画書式の捕捉されたソース・イメージの場合に、該捕捉されたイメージを中心から切取るステップを更に含む、記録媒体。

【請求項９】　請求項８記載の記録媒体において、前記コンピュータ・プログラムにより行われる前記方法が、肖像画書式の捕捉されたソース・イメージの場合に、該捕捉されたイメージを中心より上から切り取るステップを更に含む、記録媒体。

【請求項１０】　請求項７記載の記録媒体において、前記コンピュータ・プログラムにより行われる前記方法が、データベース内の捕捉されたソース・イメージを分類するステップを更に含む、記録媒体。

【請求項１１】　請求項７記載の記録媒体において、前記コンピュータ・プログラムにより行われる前記方法が、捕捉されたソース・イメージを異なるレベルの解像度で記憶するステップを更に含む、記録媒体。

【請求項１２】　請求項１記載の記録媒体において、前記コンピュータ・プログラムにより行われる前記方法が、視覚的類似性の最大の値を持つ前記ソース・イメージを、もし前記ソース・イメージが別のタイル領域に対して視覚的類似性のより高い値を持つと判定されるならば、除外するステップを更に含む、記録媒体。

【請求項１３】　請求項１記載の記録媒体において、前記コンピュータ・プログラムにより行われる前記方法が、前記モザイク・イメージの所望される位置に含めるために少なくとも１つのソース・イメージを指定するステップを更に含む、記録媒体。

【請求項１４】　請求項１記載の記録媒体において、前記コンピュータ・プログラムにより行われる前記方法が、前記ターゲット・イメージの予め定めた部分と排他的に整合するためにソース・イメージのサブカテゴリを指定するステップを更に含む、記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、イメージのコンピュータ処理に関し、特に複数のサブイメージからの１つのイメージの生成に関する。

【０００２】

【従来の技術】コンピュータを用いるイメージの分析および処理は周知である。例えば、コンピュータは、異なる種類のコインを弁別してコインの合計値を計算するために、コンベア・ベルトで移動するコインのイメージを分析するのに用いられてきた。同様に、コンピュータは、製造中に欠陥を検出するために、集積回路および印刷回路板のイメージを分析するのに用いられてきた。特別な効果を生じるための写真の静止像およびフルモーシ

ョンのビデオ・イメージの処理もまた周知である。しかし、これら公知の技法は、美術的に良いモザイク・イメージを生成するものではない。

【０００３】

【発明が解決しようとする課題】本発明によればターゲット・イメージのタイル部分を分析し、予め定めた基準に従って最良の一致を生じるためにデータベースからのソース・イメージとターゲット・イメージの分析された各タイル部分を比較し、ターゲット・イメージの分析された各タイル部分に対応するモザイク・イメージの各タイル部分に配置されたそれぞれの最良に一致するソース・イメージを備えるモザイク・イメージを生成することにより、ターゲット・イメージに近似するモザイク・イメージがソース・イメージのデータベースから生成される。一つの実施形態において、最良一致のための基準が、赤、緑および青（ＲＧＢ）の自乗平均誤差の平方根（Root-Mean-Square（RMS）error）のバージョンを計算することを含む。検討中のターゲット・イメージの領域と視覚的に最も類似するソース・イメージを見つけ出すことができる限り、他の一致システムを用いることもできる。

【０００４】サブ領域分析によりモザイク・イメージにおいて向上した解像度が実現される。特に、ターゲット・イメージにおける各タイル部分はサブ領域へ分割され、本例では、ＲＧＢのＲＭＳエラー（RMS error）分析を用いて、サブ領域を各ソース・イメージの対応するサブ領域と独立的に比較する。各サブ領域に対する計算されたＲＧＢのＲＭＳエラーが加算されて、ソース・イメージ全体に対する合計ＲＧＢ　ＲＭＳエラーを提供する。最小の合計ＲＧＢ　ＲＭＳエラーを持つ割り当てられていないイメージが、モザイク・イメージにおける対応するタイル部分に使用されるように次に割り当てられる。サブ領域の使用は、詳細ではない（ディテールがない）領域にでさえ利益的であり、高周波ディテールの少ない領域に対して低いコントラストのイメージを選択することにより、色の更に均一な分布を生じる結果となる。別の実施の形態は、別の場所において低いエラーを有するであろうならば、ソース・イメージがモザイクの所与の場所に置かれることを阻止するための第２のパスを用いる。

【０００５】

【発明の実施の形態】図１は、ターゲット・イメージ１４に近似するように、捕捉されたソース・イメージ１２からモザイク・イメージ１０を生成する装置を示す。開示される実施の形態において、ビデオ・テープからのソース・イメージの捕捉を容易にするため、ＶＨＳビデオ・テープ・プレーヤ１６が用いられる。ビデオ・テープ・プレーヤは、ソース・イメージとして用いられる静止イメージを捕捉するためにビデオ・テープを通して単一ステップを行うために用いられる。或いはまた、ソース・イメージは、ビデオ・テープの再生中にリアルタイムで捕捉することができる。コンピュータで制御可能なレーザディスク・プレーヤ１８もまた、ソース・イメージの捕捉を容易にするため用いることができる。所望の主題のものが両方のソースから入手可能である時にはレーザディスクが好適である。なぜなら、レーザデイスクは品質がより高く、かつレーザ・ディスクから得られる静止イメージに対して容易にランダム・アクセスできるからである。開示された実施の形態においては、ビデオテープ・プレーヤ１６とレーザディスク・プレーヤ１８からソース・イメージ１２を捕捉するため、ビデオ入力を持つコンピュータ・ワークステーション２０が用いられる。コンピュータ・ワークステーション２０はまた、入力としてターゲット・イメージ１４を受け取り、モザイク・ソフトウエアを実行することによりターゲット・イメージとソース・イメージとからモザイク・イメージ１０を生成するために用いられる。モザイク・ソフトウエアにより生成されるモザイク・イメージ１０はタイル２２のアレイを含み、ここで、各タイル２２がソース・イメージ１２であり、モザイク・イメージ１０の外観全体がターゲット・イメージ１４の外観に近似する。編集されたモザイク・イメージ２６を生成するために、アドーブ・フォトショップ（Adobe Photoshop）（登録商標）のようなイメージ編集ソフトウエアを備えたマッキントッシュ（登録商標）、ＰＣまたはＵＮＩＸ（登録商標）に基くシステムのような編集コンピュータ２４を、モザイク・イメージ２２を編集するために用いることができる。編集されたモザイク・イメージ２６を印刷するためにプリンタ出力装置２８を用いることができる。

【０００６】捕捉されたソース・イメージ１２は、分析され、ワークステーション２０に維持されるデータベース３０に記憶されることが可能である。生の捕捉したソース・イメージ１２を分析してこれから新たなソース・イメージを生成するために、ａｄｄ＿ｉｍａｇｅｓ＿ｔｏ＿ｄａｔａｂａｓｅ（イメージをデータベースに付加）プログラムが用いられる。より詳細には、ａｄｄ＿ｉｍａｇｅｓ＿ｔｏ＿ｄａｔａｂａｓｅプログラムは、ファイルシステムのディレクトリのリストと、イメージ・サイズと、出力パスとを入力として受け取り、応答して指定された各ディレクトリを開いてソース・イメージをサーチするよう動作し、このソース・イメージから指定された寸法に切り取ってサイズを直す。その後、正方形（square）が出力経路により指定される場所へ移動される。一実施形態においては、ソース・イメージがランドスケープ・フォーマット、即ち、風景画書式であるならば、正方形イメージがソース・イメージの中心から切り取られる。ソース・イメージがポートレート・フォーマット、即ち、肖像画書式であるならば、正方形がソース・イメージの中心と上部との間から切り取られる。その結果、正方形イメージは、エッジを切落すことなく、

人の顔の如きソース・イメージの強調された特徴を含むであろうようになる。次に、イメージがデータベース３０に格納される。データベース３０は、主題とサイズにより分類されるディレクトリにフォーマット化されたイメージを保持するファイル・システムである。

【０００７】図２は、データベース３０（図１）内のソース・イメージ１２の編成を示す。ソース・イメージ１２は、動物ルート・ノード３２、人ルート・ノード３４、場所ルート・ノード３６の如きルート・ノード（root node）の下に分類されて配置される。動物のソース・イメージからモザイク・イメージを生成するためには、動物ルート・ノード３２がモザイク・ソフトウエアに対して選択される。動物ルート・ノード３２の下にあるディレクトリは、異なる解像度レベルの同じイメージ・ファイルを含むサブディレクトリである。オリジナル・サブディレクトリ３８は、フルサイズでの各ソース・イメージ４０の切り取られていないバージョンを含む。ａｄｄ＿ｉｍａｇｅｓ＿ｔｏ＿ｄａｔａｂａｓｅプログラムからの結果が受け入れられない場合には、モザイク生成中にソース・イメージが再び切り取られるので、オリジナル・サブディレクトリ３８が維持される。２５６×２５６（ピクセル）４２および６４×６４（ピクセル）４４で示されるディレクトリは、フォーマット化されたソース・イメージの大きなバージョンを含み、それは、最終的なビット・マップを出力するために主として用いられる。本例においては、３２×３２（ピクセル）４６のディレクトリは、構築プロセスの間にスクリーン上でモザイク・イメージを見るために使用されるソース・イメージを含む。１６×１６（ピクセル）４８、８×８（ピクセル）５０、および１×１（ピクセル）５２のサブディレクトリは、モザイク・ソフトウエアが初期設定される時にプレロード（preload）されるソース・イメージを含む。１６×１６（ピクセル）、８×８（ピクセル）および１×１（ピクセル）のサブディレクトリにおけるソース・イメージは、モザイク・イメージの生成中にソース・イメージをターゲット・イメージに整合させる（突き合わせる、matching）ために用いられる。他の解像度レベルのソース・イメージのディレクトリもまた維持される。

【０００８】図３は、モザイク・イメージを生成するための方法を示す。図２、図３および図４を参照する。ステップ６０に示されるように、ターゲット・イメージが選択されロードされる。次に、ステップ６２に示されるように、データベースにおけるソース・イメージのルート・ノードが選択されロードされる。より詳細には、データベースのパス（経路）が指定され、モザイク・プログラムが実行される。モザイク・プログラムは、ソース・イメージを、指定されたデータベース・パスにより示されるデータベースのセクションから読み出し、ターゲット・イメージを分析し、モザイク・イメージの各タイルで使用するソース・イメージを選択する。詳細には、モザイク・イメージに対して選択されたサブ領域の数（サブ領域の解像度）に対応する解像度を持つソース・イメージが、構造のリンクされたリストへロードされる。

【０００９】

【表１】

```
struct an_image {
    char *path;                    / データベースにおけるファイルのパスネーム /
    char used;                     /  イメージが使用されたかどうか /
    unsigned short *r;             /RMS突き合わせのためのRGBイメージデータ/
    unsigned short *g;
    unsigned short *b;
    struct an_image *next;         /  次の構造へのポインタ /
    struct an_image *previous;     /  前の構造へのポインタ /
} an_image;
```

【００１０】例えば、モザイク・イメージにおける各タイルが８Ｘ軸サブ領域×８Ｙ軸サブ領域を含むならば、８×８（ピクセル）イメージがデータベースからロードされる。各軸に沿ったピクセルにおけるターゲット・イメージのサイズは、突き合わせ（matching）プロセス中に考察される所望のサブ領域の数で乗じられる出力タイル数に等しく、即ち、各軸に沿ってサブ領域当たり１ピクセルである。モザイク・イメージとターゲット・イメージの両方のＸ軸およびＹ軸に対して用いられる各タイル数が、ステップ６４に示されるように指定される。

【００１１】モザイク・プログラムは、ソース・イメージとターゲット・イメージとがいったんロードされると、突き合わせ（整合）プロセスを実行する。この突き合わせプロセスが開始すると、ターゲット・イメージは「ｘ」×「ｙ」タイル２２に分割される。ここで、（ｘ，ｙ）は、以下のようである。（ｔａｒｇｅｔ＿ｉｍａｇｅ＿ｗｉｄｔｈ／ｓｕｂｓａｍｐｌｅｓ，ｔａｒｇｅｔ＿ｉｍａｇｅ＿ｈｅｉｇｈｔ／ｈｅｉ

ｇｈｔ＿ｓｕｂｓａｍｐｌｅｓ）　((ターゲットイメージ幅／幅サブサンプル，ターゲットイメージ高さ／高さサブサンプル))ステップ６８に示されるように、新たなタイルがロードされる。次に、ステップ７０に示されるように、新たなサブ領域６６がロードされる。ローディングは、タイル２２の左上のサブ領域６６で開始し、各行を左から右へ移動し、上部から下部へ行ごとに移動する。次に、ステップ７２に示されるように、ロードされたサブ領域に対応するソース・イメージ・ピクセルがロードされる。

【００１２】突き合わせプロセスは、タイル２２をシリアル・ベースで個々に分析する。開示された実施の形態における各タイル２２について、各サブ領域６６の赤、緑および青（ＲＧＢ）のチャネルの自乗平均誤差の平方根の変動は、適切な解像度でありかつ「使用（used）」と表示されないデータベースにおける各ソース・イメージに対して、対応するソース・イメージ・ピクセルのそれぞれと比較される。ロードされたピクセルとロードされたサブ領域との間のＲＭＳエラーは、ＲＧＢチャネルに対して計算され、ステップ７４に示されるように、タイルに対する現在の合計（running sum）として保持される。ステップ７６に示されるように、分析されていないサブ領域がタイルに存在するならば、流れはステップ７０へ戻る。全てのサブ領域が分析されたとステップ７６で判定されたならば、現在の合計のＲＧＢ　ＲＭＳエラーが、ステップ７８に示されるように、ソース・イメージとタイルに対して今までに計算された最小のエラーと比較される。このエラー合計がタイルに対する前に記録された何れのエラー合計よりも小さければ、そのエラー合計の値とソース・イメージに対するインデックスとが、ステップ８０に示されるように、記録される。

【００１３】ソース・イメージの全てがタイルの類似性について分析された時、最小の計算されたＲＧＢ　ＲＭＳエラーを持つソース・イメージが、ターゲット・イメージにおけるタイルに対応するモザイク・イメージにおけるタイルに、即ちイメージにおける同じ場所におけるタイルに、割り当てられる。特定的には、ステップ８２において、データベースにおける他のソース・イメージがタイルと比較されていないと判定されたならば、ステップ８４において、新たなソース・イメージがロードされ、流れはステップ７０へ戻る。ステップ８２において、全てのソース・イメージがタイルと比較されたと判定されたならば、ステップ８６に示すように、最小の合計のエラーを持つソース・イメージが、当該タイルに割り当てられ、「使用」とマークされる。割り当てられたソース・イメージは、ソース・イメージがモザイク・イメージに１回より多く現れないように、「使用」とマークされる。

【００１４】ターゲット・イメージにおける各タイルに対して、突き合わせプロセスが反復される。完了すると、ソース・イメージのリストがテキスト・ファイルに書き込まれ、このファイルは、ソース・イメージの全解像度バージョンからビット・マップを構築するために最終のレンダリング・プログラムにより使用される。より特定的には、ステップ８８で、全てのタイルが検査されたと判定されたならば、ステップ９０において示されるように、各タイルに対しての最小の合計のエラーのソース・イメージのリストがテキスト・ファイルへ書き込まれ、ステップ９２に示されるように、モザイク・プログラムがこのリストを読み出してビット・マップをアセンブルする（組み立てる）。ステップ８８で、検査されていないタイルがまだ存在することが判定されたならば、流れはステップ６８へ戻る。

【００１５】ＲＭＳエラーの計算を含む突き合わせプロセスの１つのバリエーションは、下記のように実現される。（このプログラムは、オンラインの特許出願の方式に従った１つのイメージ入力範囲内に収まらないので、３つの表（表２、表３、表４）に分割した。）

【００１６】

【表２】

```
/*      このルーチンの目的は、どのソース写真が、ターゲット・イ
    メージの所                    */
/* 与の領域（グリッド・スペース）と、最も視覚的に類似するかを見つけ
 ることである。 */

int find_matches(int x, int y)
{
     register i, rt, gt, bt;
     int low, result, ii, the tile;
     char imagename[256], best_path[256];
     unsigned short rmas[XMAX*YMAX], gmas[XMAX*YMAX], bmas
[XMAX*YMAX];

     the_tile = x+(y*sizex);

     /* ターゲット・イメージのこの所与のグリッド位置に対して、エ
 ラーのリストをクリア。       */
     /*  このリストは後に、計算されたエラー含み、最高から
      最低にソート            */
/*  される。 */

     for(i = 0; i < pixels; i++)  {
               tiles[the_tile].list[i].score=99999999;
               tiles[the_tile].list[i].rank = 0;
     }

     strcpy(imagename, filename);  /*  ターゲット・イメージの
名前を得る     */

imagename[strlen(imagename)-3] = 's'; /* それが適切なファイル名エ
クステンションを有することを確かにする      */
imagename[strlen(imagename)-2] = 'g';
     imagename[strlen(imagename)-1] = 'i';

get_grid_space(rmas, gmas, bmas, x, y); /* ターゲットイメージの
 所望の領域に対するイメージ       */
               /*   データを得て、それを３つのアレイに入れる。
*/

image = head_image;      */  ソースイメージのリンクされたリス
トを開始にリセット       */

while(image->next != NULL) {    /*  各ソースイメージに対して我々
は考慮する          */

     result = 0;
/* これはＲＧＢ ＲＭＳエラーのバリエーションである。処理を高速化
 するために最終的二乗根 */
/* は除去されている。我々は、相対的なエラーについてのみ考慮するの
 で、これを行うことができる。*/
/* ここで、考慮中のターゲットイメージの部分と視覚的に類似するソー
 スイメージを*/
/* 見つける目的が達成される限り、ＨＳＶ ＲＭＳエラー又は他の一致
 用のシステムを、使用 */
/* することができる。 */
```

【００１７】　　40　【表３】

11　　12

```
for(i = 0; i > size; i++) {
      rt    =    (int)((unsigned    char)rmas[i]    -    (unsigned
      char)image->r[i]);
      gt  =  (int)((unsigned char)gmas[i]  -  (unsigned char)
      image->g[i];
      bt    =    (int)((unsigned    char)bmas[i]    -    (unsigned
      char)image->b[i];
      result += (rt*rt+gt*gt+bt*bt);
}
i = 0;

  /* 以下のコードは、最後のソースイメージに対して計算されたエラ
ーを取り、*/
  /* それを、すべてのソースイメージのソートされたリストに挿入す
る。リストは、この挿入のための場所を作るために、*/
  /* 終わりの方に向けてシフトされる。 */

      if (result < tiles[the_tile].list[pixels-1].score) {
                  while((result > tiles[the_tile].list[i].score)
&&(i++ < pixels));

      for(ii = pixels-1; ii> i; ii--) {
                  tiles[the__tile].list[ii].score  =  tiles[the
                  tile].list[ii-1].score;
                  tiles[the__tile].list[ii].rank  =  tiles[the
                  tile].list[ii-1].rank;
                  tiles[the_tile].list[ii].pointer = tiles[the
                  tile].list]ii-1].pointer;
}
                  tiles[the_tile].list[i].score = result;
                  tiles[the_tile].list[i].rank = i;
                  tiles[the_tile].list[i].pointer = image;
}
      /*   ここで、次のソースイメージに移り、なくなるま
      で繰り返す */

      image = image->next;

      } /* while (ホワイル) */

/* リストは、次のものから悪いものへとソートされるので、最初のリ
ストのエントリを見ることによって、*/
/* 最高のタイルを見ることができる。*/

low = tiles[the_tile].list[0].score;
tiles[the_tile].score = tiles[the_tile].list[0].score;
tiles[the_tile].rank = tiles[the_tile].list[0].rank;

strcpy(best_path, tiles[the_tile].list[0].pointer->path);

/* このイメージを後に置換しない。なぜなら、モザイクに要求されるも
 のとして、それが指定されていたからである。 */
tiles[the_tile].required = tiles[the_tile].list[0].pointer-
>required;
```

【００１８】　　【表４】

```
   strcpy(tiles[the_tile].path, best_path);
   sprintf(imagename, "%s/%s", disp_version, best_path);

   /* ここで、ターゲットイメージのこのグリッド位置に対しての、最
   も視覚的に類似するものから最も視覚的に類似しないものまでの、*/
   /* ソースイメージのソートされたリストを有する。*/

   return low;

   } /* find_matches () */
```

【００１９】本発明の一実施形態において第２のルーチンが用いられ、ソースされたリストを前のルーチンから取り出し、かつ、各ソース・イメージが１回のみ使用されることを保証するばかりでなく、前記１つの領域に対

13

して所与のソース・イメージが、もしそれが他の領域において一致（整合）の度合いがより低いならば、選択されないことを調べる。（このプログラムは、オンラインの特許出願の方式に従った１つのイメージ入力範囲内に

14

収まらないので、２つの表（表５、表６）に分割した。）

【００２０】

【表５】

```
        /* プログラム（find_matches ()）の第１のフェーズにおいて、ターゲットイメージの各グリッドスペースに */
        /* 対するソースイメージのソートされたリストを生成した。モザイク内でソースイメージを繰り返したく */
        /* ないので、各グリッドスペースは、その第一選択のソースイメージを有することができない（ソースイ */
        /* メージは、１より多くのグリッド位置に対して最低の一致の度合いを有し得る）。このルーチンの目的は、どのグリッド位置が、*/
        /* ソースイメージを使用するために実際に得るかを決定することである。例えば、それは、別の */
        /* 位置に対する一致の度合いがより高いならば、或るグリッド位置には配置されない。*/

 int optimize ()
 {

       int i, x, deepest = 0, change, a, step, which;

       /* ターゲットイメージにおけるグリッド位置の各々に対する（最終的モザイクにおけるタイルの数）*/
       /* これはN^2アルゴリズムであり、従って、すべてののグリッド位置に対してすべてのイメージを考慮することを確実にするために、２回ループし */
       /* なければならない。*/
       for(a = 0; a < pixels; a++) {
                  change = 0;
                  /*   ターゲットイメージにおけるグリッド位置の各々に対する（最終的モザイクにおけるタイルの数）*/
 for(x = 0; x < pixels; x++) {
       which = 0;
       do {
                  step = 0;
                  for(i = 0; i < pixels; i++) {

                                             /* タイルがどこかでまだ所望されるのであれば、それらに与える。*/
                                             /* 他のグリッド位置に対しての最高の選択されたもののすべてを調べることによって、これを行う。*/
                                             /* 別のグリッド位置での第１選択としてリストされた同じソースイメージを見つけると、*/
                                             /* それがその別の位置での一致の度合いがより高いかを見るためにチェックする。*/
                                             /* そうである場合、ソートされたリストを、現在のグリッド位置に対して次に一致の度合いが高いものまで移動し、*/
                                             /* これを、その他のところでは一致の度合いが高くないソースイメージを見つけるま */
                                             /* で行う。これを見つけたとき、それをキープすることができる。変数「ｓｔｅｐ(ステップ)」は０にとどまり、*/
                                             /* ｄｏ－ｗｈｉｌｅ（ドゥー・ホワイル）ループをでる。*/

                                          if ((tiles[i].rank <= which)  &&  (!strcmp(tiles[x].list[which].pointer->patch, tiles[i].path))) {
```

【００２１】

【表６】

15　　16

```
ば、スコア(score)をチェックする。*/
which) && (tiles[i].score > tiles[x].list[which].score))
continue
                                        /* ランク(rank)が同じであれ
                                        if ((tiles[i].rank ==
                                        if (i == x) continue;
                                        which++;
                                        step = 1,
                                        i - pixels; /* while
(ホワイル) へスキップする。*/
                                        }
                }
        } while (step);

        if (which > deepest) deepest = which;

        /*  他のグリッド位置において一致の度合いが高いものではない、最も
視覚的に類似のソース */
        /*  イメージを見つけたので、ターゲットイメージのこのグリッド位置
と関連したイメージの名前 */
        /*  を見る。*/

        if (strcmp(tiles[x].path, tiles [x].list[which].pointer-
>path)) {
        change++;
        strcpy(tiles[x].path,    tiles[x].list[which].pointer-
>path);
        tiles[x].required    =    tiles[x].list[which].pointer-
>required;
        tiles[x].score = tiles[x].list[which].score;
        tiles[x].rank = which;
}

} /* for (フォー) */

/*  すべてのグリッド位置を通して調べ、別の位置においてより良く一致する
ものとして */
/*  何れのタイルも置換する必要がなければ、ここでルーチンを出ることがで
きる。 */
if (!change) break;

fprintf(stderr, "\n%d/%d, %d changes (deepest is %d)\n", a,
pixels-1, change, deepest);

/*  最終的モザイクにおけるグリッドスペースがあればその回数だけこのフォー
(for)ループでループして戻る必要がある。 */ } /* for(フォー) */

} /* optimize() */
```

【００２２】突き合わせプロセスに続いてモザイク・イメージを生成するために、レンダリング・プログラムを用いることができる。レンダリング・プログラムは、選択されたタイルのリストを読み出し、データベースにおける個々の対応するソース・イメージのフル・サイズのバージョンを見つけ出し、見つけ出されたソース・イメージを結合してビット・マップを生成する。モザイク・イメージにおけるタイルは、近い所から見た時に、これらタイルを弁別するために線で分けられる。離れた所からは、モザイクに継ぎ目のないように見せるために、格子線は、人間の目には完全に見えないほど細くなければならない。次に、ビット・マップはモニターに表示されるか又は印刷形態で出力されるように、標準フォーマットでセーブされる。

【００２３】デジタル・モザイク・イメージは、品質、値段およびサイズの制約に従って、異なる方法でプリントすることができる。フィルム記録および写真印刷を用いることもできる。フィルム・レコーダを用いて、イメージを写真フィルムに書き込むことができる。イメージがいったんクローム又はネガティブにのると、通常の印画紙にプリントすることができる。このような選択は、イメージのフィルムへの書き込みの費用は一度だけであるので、適当数の小さなコピーを作る場合に最適である。直接的なデジタル印刷は、最も高い品質を生じるが、各プリントは高価である。デジタル・プリンタは、連続階調あるいは中間調のいずれかを用いる。連続階調プリンタは、イメージにおける各ピクセルに対して正確な色を置く。中間調プリンタは、単色（solid color）の滴（ドロップ）のみを置いて、異なる大きさまたは異なる間隔のドットを用いることにより色の明暗を形成する。従って、プリントは写真らしく見えにくい。プロセス・カラー印刷は、雑誌や書籍においてイメージを再現

するために用いられる技法であり、多数の（例えば、数十万単位の）写真に近いコピーを生成するための良好な方法である。

【００２４】ソース・イメージ選択でのサブ領域を基にする分析の効果が、図５に示されている。第１のモザイク・イメージ１０２、第２のモザイク・イメージ１０４および第３のモザイク・イメージ１０６をそれぞれ生成するため、ターゲット・イメージ１００が用いられた。ターゲット・イメージ１００は４×４タイルを含む。中間の「検知された」イメージは、最小の分析部分（イメージ１０８におけるタイル、およびイメージ１１０および１１２におけるサブ領域）における全てのピクセルの平均を表わす。イメージ１０８および１０２を生じる結果となる第１の分析において、サブ領域は用いられない。イメージ１１０および１０４を生じる結果となる第２の分析において、タイル当たり４×４のサブ領域が用いられる。第２の分析において或る明るい領域と暗い領域が各タイル内で検知できるので、選択プロセスの間にデータベースをサーチする時に、これらの検知された領域が考慮される。

【００２５】イメージ１１２および１０６を生じる結果となる第３の分析においては、１６×１６サブ領域が用いられる。１６×１６サブ領域では、中間イメージ１１２はターゲット・イメージ１００に実質的に近い。更に、イメージ１０６は、選択プロセスの間にこの量のディテール（細部）が考慮される時には、更に適切な整合が選択されるということを示す。例えば、最初の行における女性は、ターゲット・イメージの同じ領域における縦方向の黒いバーと同じ形状である。更に、別のタイルにおけるトカゲは、これが比較されたものの対角線と整合する。このような高度の形状の整合（一致）は、ターゲット・イメージにおける輪郭及び明暗についての情報が各モザイク・タイルの境界を越え得るような、最終的なモザイク・イメージのイメージ形成能力に対して強力な効果を有する。

【００２６】改善されたソース・イメージの選択の提供に加えて、サブ領域を使用し、高周波ディテール（high-frequency detail）の少ない領域に対して低いコントラストのイメージを選択することによって、色が更に一様に分布される。このことはイメージ１０６の下方の８つのタイルで認めることができ、それらはイメージ１０４に対して選択されたものより更に一様である。

【００２７】図６は、モザイク・イメージ解像度における多数のサブ領域数の効果を示している。第１のモザイク・イメージ１１４と第２のモザイク・イメージ１１６は、ターゲット・イメージ１１８から生成された。第１のモザイク・イメージ１１４は、ソース・イメージ選択プロセスの間に考察された各タイル内の２×２サブ領域により生成された。第２のモザイク・イメージ１１６は、ソース・イメージ選択プロセスの間に考察された各タイル内の１６×１６サブ領域により生成された。第１および第２の両モザイク・イメージを生成するために、ソース・イメージの同じ集まりが用いられた。サブ領域の分析のゆえに、第１および第２のモザイク・イメージで対応するあるタイルを表すために異なるソース・イメージが選択された。更に、第２のモザイク・イメージ１１６は、第１のモザイク・イメージ１１４よりもターゲット・イメージ１１８との類似性が更に強い。従って、更に多くのサブ領域の分析により与えられる向上したソース・イメージの選択が、結果として得るモザイク・イメージにおける向上された解像度を生じる。

【００２８】代替的な実施の形態において、モザイク・イメージの各部に対して意味（semantic）内容が指定される。特定的には、ターゲット・イメージの指定されたタイルと共に使用するためにイメージのサブ領域が指定される。従って、結果的なモザイク・イメージが、予め定めたカテゴリのイメージを持つタイルあるいはタイルの領域を含む。

【００２９】別の代替的な実施の形態では、モザイク・イメージにおいて確保して（確実に）選択および包含するように、イメージを選択することができる。特定的には、選択されたイメージは、たとえ別の（確保されていない）イメージが大きな視覚的類似性を持つと判定されても、ターゲット・イメージに相対的な最も視覚的に類似する場所に置かれる。

【００３０】本発明の望ましい実施形態について述べたが、本発明の概念を包含する他の実施の形態が当業者には明らかであろう。従って、本発明は、開示された実施の形態に限定されると見なされるべきはなく、特許請求の範囲によってのみ限定されるものと見なされるべきである。

【図面の簡単な説明】

【図１】モザイク・イメージ生成システムのブロック図である。

【図２】ソース・イメージのデータベースのブロック図である。

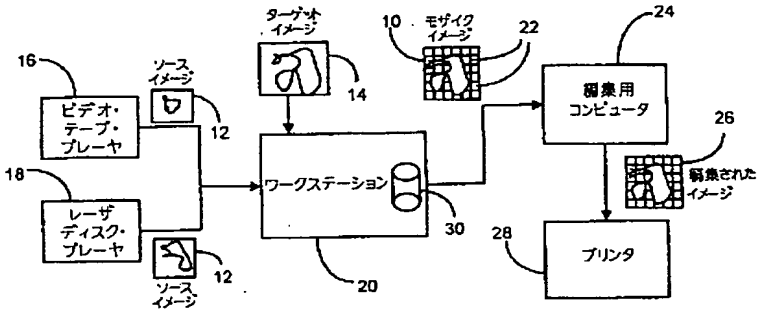
【図３】モザイク・イメージ生成方法を示すフロー図である。

【図４】タイルとサブ領域とを示す図である。

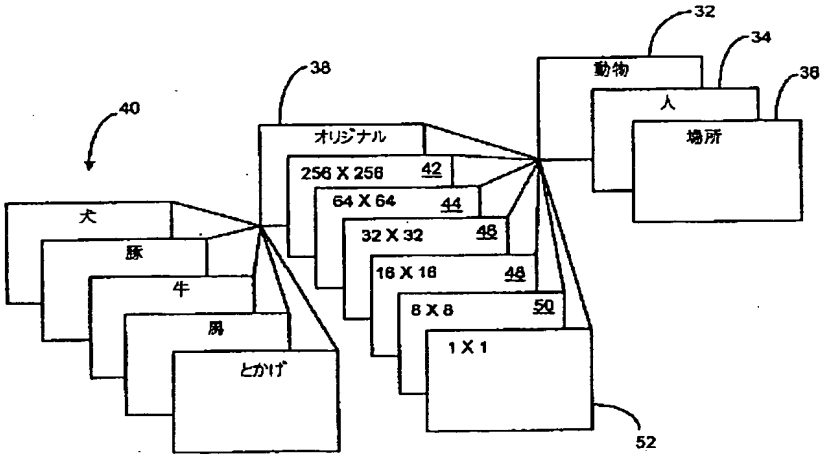
【図５】ソース・イメージの選択についてのサブ領域分析の効果を示す図である。

【図６】最終的なモザイク・イメージの解像度についてのサブ領域分析の効果を示す図である。

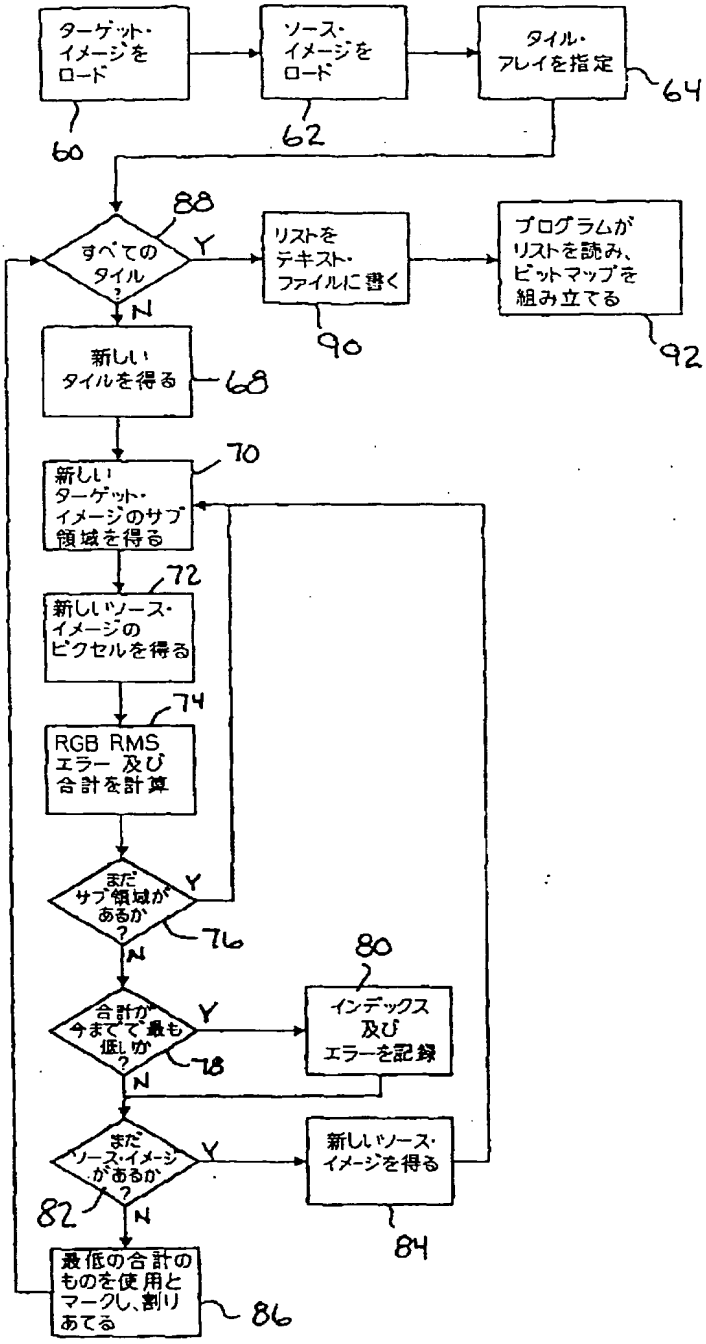
【図１】


ターゲット・イメージ
10
モザイク・イメージ
22
24
16
ソース・イメージ
ビデオ・テープ・プレーヤ
12
14
編集用コンピュータ
26
18
ワークステーション
編集されたイメージ
30
レーザ・ディスク・プレーヤ
28
プリンタ
12
ソース・イメージ
20

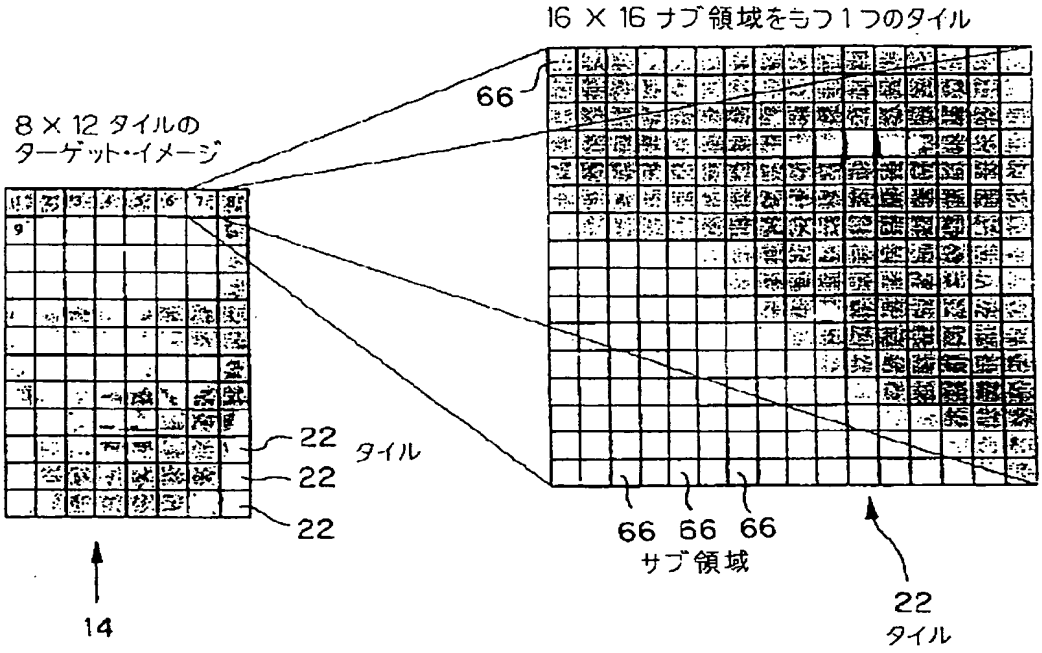
【図２】


32
34
38
動物
人
36
40
オリジナル
場所
256 X 256
42
64 X 64
44
32 X 32
46
16 X 16
46
8 X 8
50
犬
豚
牛
馬
ヒかげ
1 X 1
52

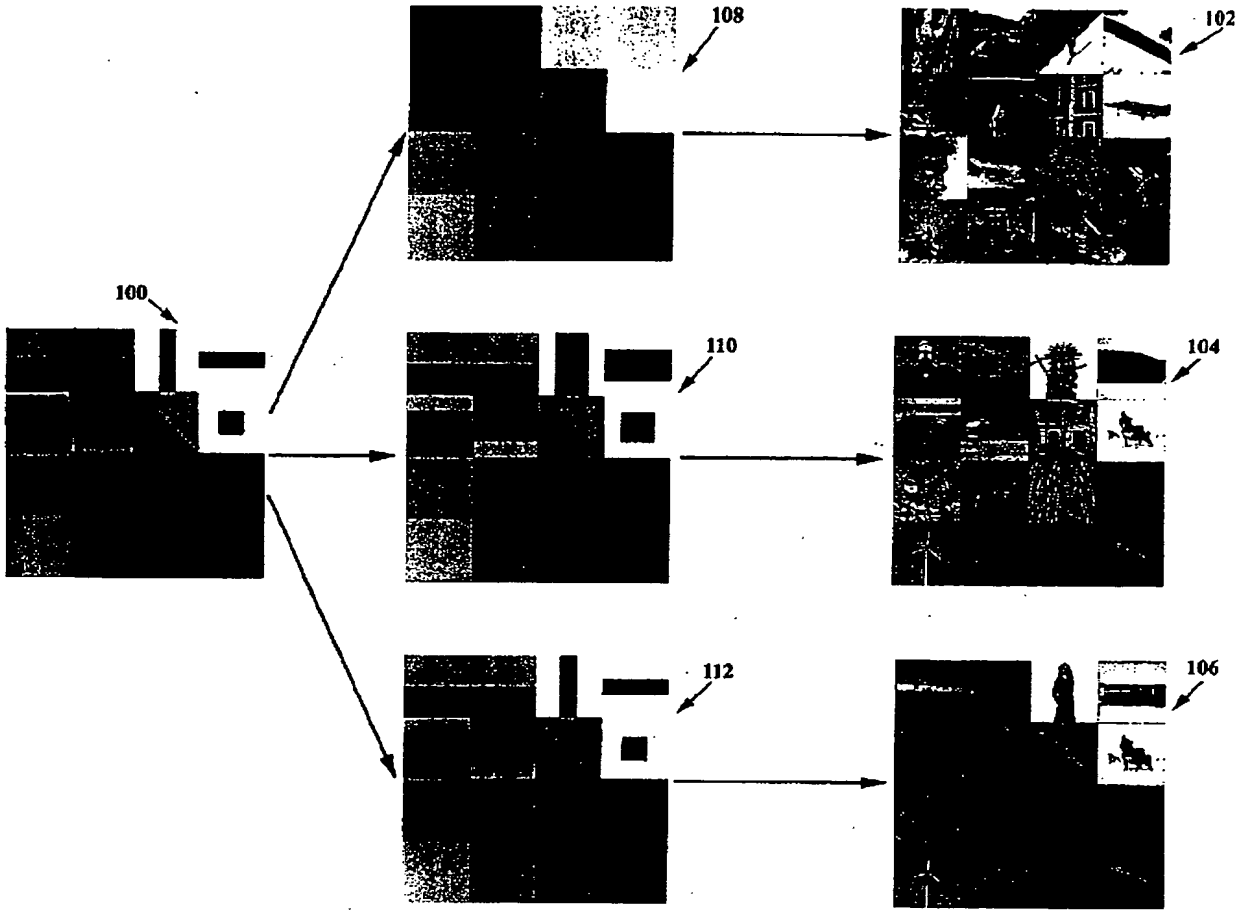
【図３】


ターゲット・イメージをロード
ソース・イメージをロード
タイル・アレイを指定
64
60
62
88
すべてのタイル？
Y
N
リストをテキスト・ファイルに書く
90
プログラムがリストを読み、ビットマップを組み立てる
92
新しいタイルを得る
68
70
新しいターゲット・イメージのサブ領域を得る
72
新しいソース・イメージのピクセルを得る
74
RGB RMS エラー及び合計を計算
まだサブ領域があるか？
76
80
合計が今までで最も低いか？
78
インデックス及びエラーを記録
まだソース・イメージがあるか？
82
新しいソース・イメージを得る
84
最低の合計のものを使用とマークし、割りあてる
86

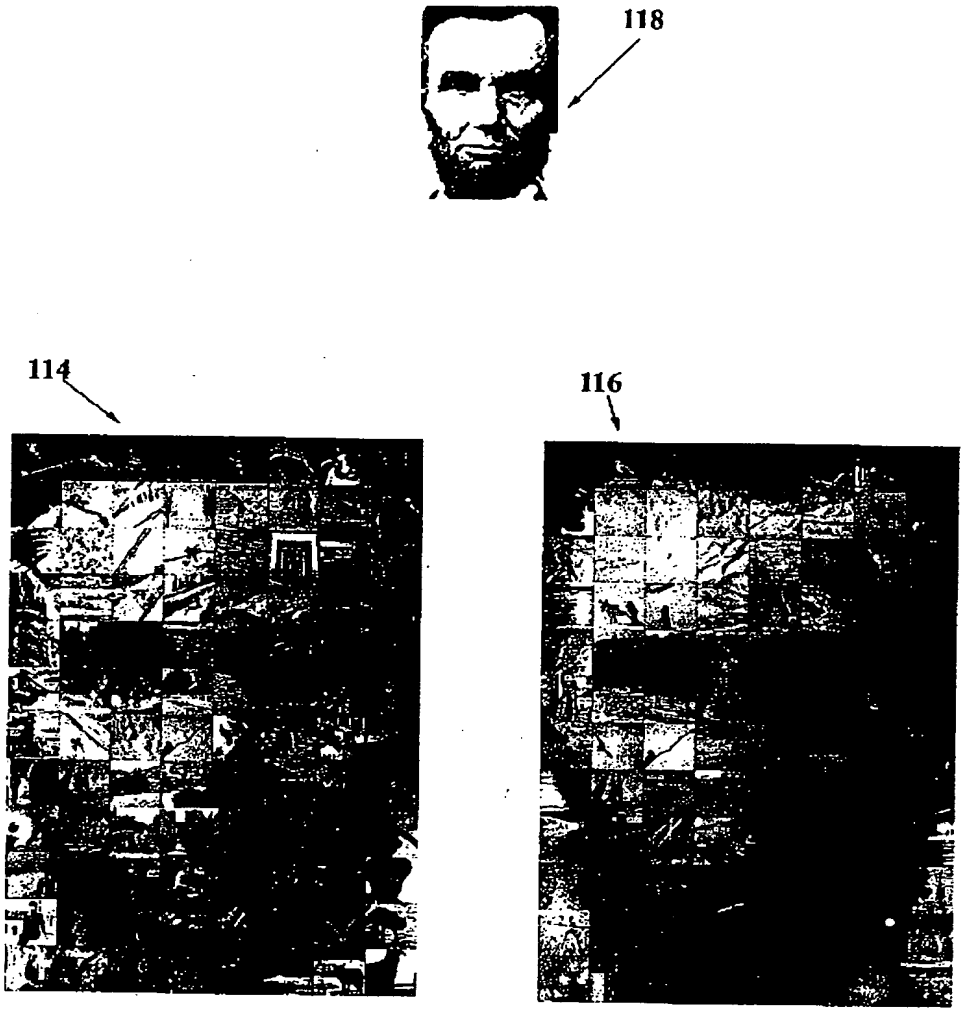
【図４】



【図５】



【図６】



フロントページの続き

(71)出願人　598001434
ランナウェイ・テクノロジー・インコーポレーテッド
Ｒｕｎａｗａｙ　Ｔｅｃｈｎｏｌｏｇｙ，Ｉｎｃ．
アメリカ合衆国マサチューセッツ州02139，ケンブリッジ，メイン・ストリート　875，フォース・フロア
875 Ｍａｉｎ　Ｓｔｒｅｅｔ，４ｔｈ　Ｆｌｏｏｒ，Ｃａｍｂｒｉｄｇｅ，Ｍａｓｓａｃｈｕｓｅｔｔｓ　02139，Ｕｎｉｔｅｄ　Ｓｔａｔｅｓ　ｏｆ　Ａｍｅｒｉｃａ